Формальные грамматики

Глава 10. Сопоставление с образцом.

10.1. Простейший пример.

10.1.1. Имеется последовательность символов x[1]..x[n]. Определить, имеются ли в ней идущие друг за другом символы "abcd".(Другими словами,

требуется выяснить, есть ли в слове x[1]..x[n] подслово "abcd".

Решение. Имеется примерно n (если быть точным, n-3) позиций,на которых может находиться искомое подслово в исходном слове.Для каждой из

позиций можно проверить, действительно ли там оно находится, сравнив четыре символа. Однако есть более эффективный способ. Читая словох[1]..x[n]

слева направо, мы ожидаем появления буквы 'a'. Как только она появилась, мы ждем за ней букву'b', затем 'c', и, наконец, 'd'. Если наши ожидания

оправдываются, то слово "abcd" обнаружено. Если же какая-то из нужных букв не появляется, мы оказываемся у разбитого корыта и начинаем

всесначала.

Этот простой алгоритм можно описать в разных терминах. Используя терминологию так называемых конечных автоматов, можносказать, что при чтении

слова x слева направо мы в каждый момент находимся в одном из следующих состояний: "начальное" (0),"сразу после a" (1), "сразу после ab" (2),

"сразу после abc" (3)и "сразу после abcd" (4). Читая очередную букву, мы переходим в следующее состояние по правилу

Текущее ***Очередная*** ***Новое***

состояние ***буква*** ***состояние***

0 ***a*** ***1***

0 ***кроме a*** ***0***

1 ***b*** ***2***

1 ***a*** ***1***

1 ***кроме a,b*** ***0***

2 ***c*** ***3***

2 ***a*** ***1***

2 ***кроме a,c*** ***0***

3 ***d*** ***4***

3 ***a*** ***1***

3 ***кроме a,d*** ***0***

Как только мы попадем в состояние 4, работа заканчивается.

Соответствующая программа очевидна:

```
i:=1; state:=0;

{i - первая непрочитанная буква, state - состояние}

while (i<> n+1) and (state <> 4) do begin

if state = 0 then begin

if x[i] = a then begin

state:= 1;

end else begin

state:= 0;

end;

end else if state = 1 then begin

if x[i] = b then begin
```

state:= 2;

end else if x[i] = a then begin

state:= 1;

end else begin

state:= 0;

end;

end else if state = 2 then begin

if x[i] = c then begin

state:= 3;

end else if x[i] = a then begin

state:= 1;

end else begin

```
state:= 0;

end;

end else if state = 3 then begin

if x[i] = d then begin

state:= 4;

end else if x[i] = a then begin

state:= 1;

end else begin

state:= 0;

end;

end;
```

end;

answer:= (state = 4);

Иными словами, мы в каждый момент храним информацию о том, какое максимальное начало нашего образца "abcd" является концом прочитанной части.(Его длина и есть то "состояние", о котором

шла речь.)

Терминология, нами используемая, такова. Слово - это любая последовательность символов из некоторого фиксированного конечного множества. Это

множество называется алфавитом, его элементы

- буквами. Если отбросить несколько букв с конца слова, останется другое слово, называемое началом первого. Любое слово также считается своим

началом. Конец слова - то, что останется, если отбросить несколько первых букв. Любое слово считается своим концом. Подслово - то, что останется,

если отбросить буквы и сначала, и с конца. (Другими словами, подслова - это концы начал,или, что то же, начала концов.)

В терминах индуктивных функций (см. раздел 1.3) ситуациюможно описать так: рассмотрим функцию на словах, которая принимает два значения

"истина" и "ложь" и истинна на словах, имеющих

"abcd" своим подсловом. Эта функция не является индуктивной, но имеет индуктивное расширение

x ->длина максимального начала слова abcd, являющегося концом x

---

10.2. Повторения в образце - источник проблем.

10.2.1. Можно ли в предыдущих рассуждениях заменить слово "abcd" на произвольное слово?

Решение. Нет, и проблемы связаны с тем, что в образце могут быть повторяющиеся буквы. Пусть, например,мы ищем вхождения слова "ababc". Вот

появилась буква "a", за ней идет "b", за ней

идет "a", затем снова "b". В этот момент мы с нетерпением ждембуквы "c". Однако - к нашему разочарованию - вместо нее появляется другая буква, и

наш образец "ababc" не обнаружен. Однако

нас может ожидать утешительный приз: если вместо "c" появилась буква "a", то не все потеряно: за ней могут последовать буквы "b" и "c", и

образец-таки будет найден.

Вот картинка, поясняющая сказанное:

x□□ y□□ z□□ a□□ b□□ a□□ b□□ a□□ b□□ c□□ ....□ <- входное слово

a□□ b□□ a□□ b□□ c□□□□□□ <-□ мы ждали образца здесь

a□□ b□□ a□□ b□□ c□ <-□ а он оказался здесь

Таким образом, к моменту

|

x□□ y□□ z□□ a□□ b□□ a□□ b /□□□□□□□□□□□□□□ <- входное слово

/

a□□ b□□ a□□ b | c□□□ □□□<-□ мы ждали образца здесь

/

a□□ b | a□□ b□□ c□ <-□ а он оказался здесь

/

есть два возможных положения образца, каждое из которых подлежит проверке. Тем не менее по-прежнему возможен конечный автомат, читающий

входное слово буква за буквой и переходящий из состояния в состояние в зависимости от прочитанных букв.

10.2.2. Указать состояния соответствующего автомата и таблицу перехода (новое состояние в зависимости от старого и читаемой буквы).

Решение. По-прежнему состояния будут соответствовать наибольшему началу образца, являющемуся концом прочитанной частислова. Их будет шесть:

0, 1 ("a"), 2 ("ab"), 3 ("aba"), 4("abab"), 5 ("ababc"). Таблица перехода:

Текущее ***Очередная*** ***Новое***

состояние ***буква*** ***состояние***

0 ***a*** ***1 (a)***

0 ***кроме a*** ***0***

1 (a) ***b*** ***2 (ab)***

1 (a) ***a*** ***1 (a)***

1 (a) ***кроме a,b*** ***0***

2 (ab) ***a*** ***3 (aba)***

2 (ab) ********** ***кроме a*** ********** ***0***

3 (aba) ********** ***b*** ********** ***4 (abab)***

3 (aba) ********** ***a*** ********** ***1 (a)***

3 (aba) ******** ***кроме a,b*** ******* ***0***

4 (abab) ********** ***c*** ********** ***5 (ababc)***

4 (abab) ********** ***a*** ********** ***3 (aba)***

4 (abab) ******* ***кроме a,c*** ******* ***0***

Для проверки посмотрим, к примеру, на вторую снизу строку. Если прочитанная часть кончалась на "abab", а затем появилась буква"a", то теперь

прочитанная часть кончается на "ababa". Наибольшее начало образца ("ababc"), которое есть ее конец - это

"aba".

Философский вопрос: мы говорили, что трудность состоит втом, что есть несколько возможных положений образца, каждое из которых может

оказаться истинным. Им соответствуют несколько начал образца, являющихся концами входного слова. Но конечный автомат помнит лишь самое длинное

из них. Как же остальные?

Философский ответ. Дело в том, что самое длинное из них определяет все остальные - это его концы, одновременно являющиеся его началами.

Не составляет труда для любого конкретного образца написать программу, осуществляющую поиск этого образца описанным способом. Однако хотелось

бы написать программу, которая ищет произвольный образец в произвольном слове. Это можно делать в два этапа: сначала по образцу строится

таблица переходов конечного автомата, а затем читается входное слово и состояние преобразуется в соответствии с этой таблицей. Подобный метод

часто используется для более сложных задач поиска (см. далее), но для поиска подслова существует более простой и эффективный алгоритм,

называемый алгоритмом Кнута - Морриса - Пратта. Но прежде нам

понадобятся некоторые вспомогательные утверждения.

10.3. Вспомогательные утверждения

Для произвольного слова X рассмотрим все его начала, одновременно являющиеся его концами, и выберем из них самое длинное,(Не считая, конечно, самого слова X) Будем обозначать его n(X).

Примеры: n(aba)=a, n(abab)=ab, n(ababa)=aba, n(abc) = пустое слово.

10.3.1. Доказать, что все слова n(X), n(n(X)), n(n(n(X)))и т.д. являются началами слова X.

Решение. Каждое из них (согласно определению) является началом предыдущего.

По той же причине все они являются концами слова X.

10.3.2. Доказать, что последовательность предыдущей задачи обрывается (на пустом слове).

Решение. Каждое слово короче предыдущего.

Задача. Доказать, что любое слово, одновременно являющееся началом и концом слова X (кроме самого X) входит в последовательность n(X), n(n(X)),...

Решение. Пусть слово Y есть одновременно начало и конец X. Слово n(X) - самое длинное из таких слов, так что Y не длиннее n(X). Оба эти слова

являются началами X, поэтому более короткое

из них является началом более длинного: Y есть начало n(X). Аналогично, Y есть конец n(X). Рассуждая по индукции, можно предполагать, что

утверждение задачи верно для всех слов короче X, в частности, для слова n(X). Так что слово Y, являющееся концом и началом n(X), либо равно n(X),

либо входит в последовательность n(n(X)), n(n(n(X))), ..., что и требовалось доказать.